

Having thus described our invention what we claim as new and desire to secure by Letters Patent is:

Claims

1. A computer graphics processor system having the capability of mapping texture onto a three dimensional object in a scene being displayed, the system comprising:

a texture address calculator for generating texel addresses for a list of primitives being processed;

a texture main memory containing an array of texels, each texel having an address and one of N identifiers;

a texture cache memory having addresses partitioned into N banks, each bank containing texels transferred from said main memory that have the corresponding identifier;

a texture cache controller for determining and requesting the necessary transfer of texels from said texture main memory addresses to said texture cache memory addresses; and

a texture cache arbiter for scheduling controlling the actual transfer of texels from said texture main memory into the texture cache memory and controlling the outputting of texels for each pixel to a interpolating filter from the cache memory.

545
AB 2. The system of claim 1, wherein the system further includes a texture addressing scheme for organizing the array of texels in main memory to group spatially related texels in one memory page.

Sub 3. The system of claim 2, wherein the system further includes a span based polygon rasterization scheme so neighboring pixels of a primitive will be processed sequentially.

4. The system of claim 3, wherein the texture mapping capability includes storing pre-filtered texture maps at different resolutions and bilinear interpolation texture filtering.

5. The system of claim 2, wherein said texture main memory contains an array of texels having addresses arranged in rows and columns, there being a plurality of even numbered rows and columns and a plurality of odd numbered rows and columns of texels, said texels having a per texel cache memory identifier attached to each address in accordance with the following criteria: a first identifier being assigned to texels that have addresses in both even rows and

even columns of said main memory; a second identifier being assigned to texels that have addresses in both even rows and odd columns of said main memory, a third identifier being assigned to texels that have addresses in both odd rows and even columns of said main memory, and a fourth identifier being assigned to texels having addresses in both odd rows and odd columns of said main memory.

6. The system of claim 5, wherein said texture cache memory is arranged in four banks of memory in accordance with the following criteria: a first bank containing texels having the first identifier; a second bank containing texels having the second identifier; a third bank containing texels having the third identifier; and a fourth bank containing texels having the fourth identifier.

7. The system of claim 1, wherein N is equal to four and said texture main memory is organized into a plurality of texel blocks each having one of four block texel cache memory identifier in accordance with the following criteria: each texel block consisting of at least one group of four contiguous texels, the texels in each group consisting of one of each of the per texel cache memory identifiers, and wherein said texture cache memory being partitioned into a plurality of rows corresponding to said plurality of block texel cache memory identifiers, each cache

memory bank having at least one row corresponding to each of the four block texel cache memory identifiers.

8. The system of claim 1, wherein said cache controller includes N stages.

9. The system of claim 7, wherein said cache controller includes four stages, each stage controlling the transfer of texels for one of the four block texel cache memory identifiers.

10. The system of claim 9, wherein the cache controller transfers texture data at the main memory access granularity.

11. The system of claim 10, wherein each of said four stages is a least recently used controller coupled in succession thereby transferring texels according to a least recently used replacement algorithm.

12. The system of claim 11, wherein the cache controller pre-fetches necessary neighboring texels from main memory for the bilinear texture filtering.

13. The system of claim 9, wherein the texture cache memory is a multi-ported cache memory enabling multiple texel accesses per clock.

Sub
A10

14. The system of claim 1, wherein the cache read/write arbiter is coupled between said cache controller and said texture cache memory for determining which texels in the cache memory can be overwritten when new texels are determined to be transferred to said cache memory by said cache controller.

15. The system of claim 14, wherein said cache read/write arbiter transfers data according to a look-ahead algorithm to hide read and write access clocks between sequential pixels.

Sub
CH

16. The system of claim 7, wherein said texel blocks in said main memory each consist of a double quad word of data.

17. The system of claim 7, wherein each row of said cache memory consisting of four sub-rows of data, each sub-row consisting of a pair of an even sub-row and an odd sub-row, each double quad word being stored in one pair of said even and odd sub-row of said cache memory.

18. A method of controlling the transfer of texture data between a texture main memory and a texture cache memory while maintaining the most recently used data in the texture cache memory comprising the steps of:

ADD
A11

h

(a) receiving texture addresses for a first pixel, checking if the addresses match the addresses in a first stage of a multi-stage cache controller and doing one of the following, (1) loading the addresses in the first stage if there is no valid address in the first stage (2) reloading the addresses in the first stage if a match is found or (3) moving to a second stage if no match is found;

(b) if step (a)(1) is true transferring the corresponding texture data from main memory into cache memory with a first tag;

(c) if step (a)(2) is true, making no transfer of texture data because data has already been transferred;

(d) if step (a)(3) is true, checking if the addresses match the addresses in the second stage and doing one of the following (1) if there is no addresses in the second stage moving the addresses from the first stage to the second stage and loading the addresses into the first stage (2) if a match is found moving the addresses from the first stage to the second stage and loading the addresses into the first stage (3) moving to a third stage if no match is found;

(e) if step (d)(1) is true transferring corresponding texture data from the main memory into the cache memory with a second tag;

(f) if step (d)(2) is true making no transfer of texture data because data has already been transferred;

(g) if step (d)(3) is true, repeating step (d) for subsequent stages and using subsequent tags where necessary, until a last stage been checked or until a match has been found;

(h) if the last stage has been checked and no match found loading the addresses into the first stage and moving the stored addresses to the next stage in sequence and overwriting the addresses from the last stage; and

(i) if step (h) is true transferring corresponding texture data from the main memory into cache memory with the tag of the last stage addresses;

(j) wherein when addresses are loaded into the first stage the tag assigned will be either the tag of the last stage or the tag within the stage that was hit.